Upstreaming jank-lang Patches to CppInterOp Final presentation

Iva Fezova
Mentor: Vasil Vasilev

Goals

Understand the upstream workflow

Learn how compiler-research/CppInterOp accepts contributions from forks like jank-lang

Identify new API functions

Audit the jank-lang fork for functions not yet present in upstream CppInterOp

Resolve naming conflicts

Check if equivalent functionality already exists under a different name before adding anything new

Upstream new utility functions

Port new integral type transform functions from jank-lang, following the upstream coding conventions

Write unit tests

Add test coverage in TypeReflectionTest.cpp covering all integral type cases including __int128

Background

CppInterOp

A C++ interoperability library built on top of Clang/LLVM.

Provides a stable API for language runtimes to introspect and call into C++ code without writing Clang internals directly.

Maintained by the compiler-research group.

jank-lang/CppInterOp fork

jank is a Clojure dialect that compiles to native code via Clang/LLVM.

Its author (jeaye) maintains a fork of CppInterOp with many new API additions needed to implement jank features.

Goal: upstream the useful, general-purpose additions back into compiler-research.

Auditing the Fork

44 commits ahead of upstream - classified by complexity and risk:

Tier 1 — 7 commits

Config / Cleanup

Remove workflow, build fixes, null returns

Tier 2 — 17 commits

New Predicates & Transforms

IsConstType, GetVoidType, GetSignedType...

Tier 3 — 13 commits

New Behaviour

RTTI, narrowing conversions, operator[]

Tier 4 — 7 commits

Large Reworks - LLVM Porting

Overload resolution, LLVM 21/22 port

Many Tier 1 & 2 functions already existed upstream under different names (e.g. GetLValueReferenceType -> GetReferencedType)

What Was Added

After deduplication against existing upstream API, two functions were upstreamed:

GetSignedType

```
TCppType_t GetSignedType(TCppType_t type)
```

Returns the signed counterpart of an integral type.
e.g. unsigned int -> int
unsigned long -> long
float -> float (unchanged)

GetUnsignedType

```
TCppType_t GetUnsignedType(TCppType_t type)
```

Returns the unsigned counterpart of an integral type.
e.g. int -> unsigned int
long -> unsigned long
float -> float (unchanged)

Implementation

Upstream uses TableGen (.td files) to auto-generate API declarations - different from the jank fork which has hand-written headers.

1. **`CppInterOp.td`**

   Add a def block for each new function - TableGen generates the declaration in CppInterOpDecl.inc automatically

2. **`CppInterOp.cpp`**

   Add the implementation following upstream conventions: INTEROP_TRACE, null check, INTEROP_RETURN

3. **`Build`**

   cd build && make -j12 - verify the .inc file picks up the new declarations

4. **`TypeReflectionTest.cpp`**

   Add TYPED_TEST cases covering all integral types, edge cases, and nullptr inputs

Test Coverage

Unsigned -> Signed

unsigned char, unsigned short, unsigned int, unsigned long, unsigned long long, unsigned __int128

Already Signed

signed char, short, int, long, long long, __int128 - returned unchanged

Signed -> Unsigned

All signed integral types converted to their unsigned counterparts

Already Unsigned

All unsigned types returned unchanged

Non-integral types

float - returned unchanged (no signed/unsigned concept)

Null input

GetSignedType(nullptr) and GetUnsignedType(nullptr) both return nullptr

Challenges

File path differences

The jank fork uses include/clang/Interpreter/CppInterOp.h and lib/Interpreter/CppInterOp.cpp. Upstream renamed these. Cherry-pick fails - changes must be applied manually.

TableGen API generation

Upstream auto-generates declarations from .td files. The jank fork has hand-written headers. Each new function needs a def block in CppInterOp.td, not just a header declaration.

Duplicate functionality

Several functions from jank already exist upstream under different names (GetLValueReferenceType -> GetReferencedType, GetTypeWithConst -> AddTypeQualifier). Required careful API audit before adding anything.

Test environment

Missing libPolly.a prevented running tests via make check-cppinterop. Tests were validated indirectly through successful compilation and CI on the PR.

Moving Forward

-> **Continue upstreaming Tier 2 functions**

Work through remaining simple predicates and type transforms from the jank fork that don't duplicate existing upstream API

-> **Write tests per PR**

Each upstreamed function gets its own or grouped PR with unit tests - keeping reviews small and focused

-> **Tackle Tier 3 next**

Functions like MangleRTTI, IsCStyleConvertible, and narrowing conversion support after Tier 2 is complete

-> **Defer Tier 4**

LLVM 21/22 porting and overload resolution rework require coordination with mentors and broader testing

Any Questions?